Approximate energy-efficient encoding for serial interfaces

Approximate Energy-Efficient Encoding for Serial Interfaces

DANIELE JAHIER PAGLIARI, Dipartimento di Automatica e Informatica, Politecnico di Torino
ENRICO MACII, Dipartimento di Automatica e Informatica, Politecnico di Torino
MASSIMO PONCINO, Dipartimento di Automatica e Informatica, Politecnico di Torino

Serial buses are ubiquitous interconnections in embedded computing systems that are used to interface processing elements with peripherals, such as sensors, actuators and I/O controllers. In spite of their limited wiring, as off-chip connections they can account for a significant amount of the total power consumption of a system-on-chip device. Encoding the information sent on these buses is the most intuitive and affordable way to reduce their power contribution; moreover, the encoding can be made even more effective by exploiting the fact that many embedded applications can tolerate intermediate approximations without a significant impact on the final quality of results, thus trading off accuracy for power consumption.

We propose a simple yet very effective approximate encoding for reducing dynamic energy in serial buses. Our approach uses *differential encoding* as a baseline scheme, and extends it with bounded approximations to overcome the intrinsic limitations of differential encoding for data with low temporal correlation. We show that the proposed scheme, besides yielding extremely compact codecs, is superior to all state-of-the-art approximate serial encodings over a wide set of traces representing data received or sent from/to sensor or actuators.

CCS Concepts: •**Hardware** → **Buses and high-speed links; Interconnect power issues;** *Logic synthesis; Sensors and actuators; Metallic interconnect; Application specific integrated circuits;* •**Computer systems organization** → *Embedded hardware;*

Additional Key Words and Phrases: Low Power; Interfaces; Serial Communication; Bus Encoding; Approximate Computing

1. INTRODUCTION

Serial off-chip communication links are a de-facto standard for interconnecting I/O peripherals in digital systems due to a number of advantages over parallel interfaces. Thanks to the absence of skew, crosstalk, and jitter issues they allow higher signaling frequencies than their parallel counterparts. Moreover, they result in cheaper implementations thanks to the lower pin count and easier routing [Vahid and Givargis 2001]. Off-chip serial buses are most commonly used to connect processing elements (e.g., SoCs) with sensors, actuators and I/O controllers, in most cases using standard protocols such as *SPI*, *I2C*, *CAN*, etc.

Even if consisting of a single data line, serial buses can still be significant contributors to the total energy budget of a device. In fact, these buses are typically implemented as PCB traces (e.g., microstrips), whose capacitances have not scaled with the same trend as on-chip interconnects. The energy consumption of a PCB trace is in the order of 1-2 pJ/bit/inch [Stanley-Marbell and Rinard 2016; Alfke 2008]. Considering that a typical trace spans several centimeters, the transmission of a single bit can easily require around 10 pJ. This figure should be compared to the cost of executing an instruction in a micro-controller for sensor-based systems, which have active currents in the order of 50-100 $\mu A/MHz$, translating to 10-20 pJ/instruction for the typical operating frequencies [Bracke et al. 2007; Ickes et al. 2008] As a rule of thumb, we can thus *roughly equate the transmission of each single bit on a bus with the execution of one instruction*. Moreover, we should consider that in a complex system there can easily be ten or more sensors operating concurrently.

The main source of consumption incurred transmitting data in off-chip connections is dynamic energy due to transitions between electrical levels, that correspond to bit toggles at the logic level. Therefore, as widely explored in the past for parallel buses, most strategies proposed in literature to save energy in serial connections focus on

encoding the data sent on the bus with the objective of reducing the number of bit toggles [Lee et al. 2004; Ghosh et al. 2014; Salerno et al. 2004; Ren et al. 2012; Zeng et al. 2014; Poncino and Macii 2006; Stanley-Marbell and Rinard 2016; Jahier Pagliari et al. 2016a; 2016b].

Most encoding algorithms for off-chip serial buses are *lossless*, meaning that the input data can be recovered exactly at the decoder's end, provided that there are no channel errors [Lee et al. 2004; Ghosh et al. 2014; Salerno et al. 2004; Ren et al. 2012; Zeng et al. 2014]. However, serial buses are often involved in interfacing devices that are *error resilient*, i.e., that can tolerate some approximations without a significant impact on the final output quality [Chippa et al. 2013]. This property can be exploited to construct a *lossy* encoding, that can reach a larger energy saving at the expense of non-exact decoding. For instance, error resilience is typical in applications that process data sampled from analog sensors. In this case, approximations can be tolerated as long as they are negligible with respect to the environmental noise and A/D conversion errors. Similarly, resilience is also present when outputs are intended for human consumption, as in multimedia, where the limited perception of human sense organs allows small approximations without a sensible quality loss.

The idea of accepting approximations to improve other design metrics (in particular energy efficiency) has been formalized in the Approximate Computing paradigm [Han and Orshansky 2013]. Most techniques in this context focus on approximating *computations*, but the same ideas have been recently applied also to *communication* interfaces [Stanley-Marbell and Rinard 2016; Jahier Pagliari et al. 2016a; 2016b].

One very simple yet effective lossless encoding for serial buses is the one proposed by [Lee et al. 2004], which is based on word-level *differential encoding (DE)*, i.e., sending on the bus the bitwise differences between subsequent words. The effectiveness of this method is due to the strong *correlation* between subsequent words in many realistic input sets. However, pure differential encoding may incur in an increase of the total energy when data exhibit poor correlation. In this paper, which extends the work of [Jahier Pagliari et al. 2016a], we propose a new set of difference-based encodings that, by allowing approximations, overcomes the limitations of lossless DE. More specifically, the following are the main contributions of our work:

— We define **a variant of lossless DE called ADE** (Approximate DE), first introduced in [Jahier Pagliari et al. 2016a], which achieves significantly larger energy saving with respect to state-of-the-art serial encodings (both accurate and approximate), with minimal impact on quality and negligible usage of hardware resources;
— We present a set of **variants of the basic ADE**, in order to account for scenario-specific aspects (e.g., preserving small variations, balancing 0's and 1's in the stream, etc.) which could not be managed by the original encoding. We thus propose a **complete framework** for approximate difference-based encodings, from which a designer can select a particular variant based on his/her requirements, and we assess its applicability to real industrial protocols;
— We propose **an integrated co-optimization of the encoder and the analog-to-digital converter' (ADC)** used in sensor interfaces, for improved energy efficiency;
— We **compare ADE with most state-of-the-art lossless and lossy serial encodings**, using realistic bus traces generated by error resilient applications, as well as random data with different distributions.
— We provide **an accurate assessment of the effectiveness of ADE** by accounting the actual tradeoff between the hardware implementation of the codec and the energy saved in the interconnect.

The manuscript is organized as follows: Section 2 provides some basic background on the topic and surveys the state of the art in lossless and lossy low-power serial bus

encodings. Section 3 illustrates the basic operations of DE and shows its limitations, which constitute the starting point of ADE. Section 4 describes the functional and implementation details of ADE, while Section 5 presents some enhancements to the basic ADE and Section 6 describes the encoder/ADC co-optimization. Section 7 presents an assessment of the effectiveness of ADE, both in terms of absolute reduction of bus transitions and of energy/quality tradeoff, as well as a precise quantification of the overhead using accurate PCB trace data. Section 8 provides a few concluding remarks.

2. BACKGROUND AND RELATED WORK

Due to the large pitch of PCB lines, off-chip serial connections can be considered as purely capacitive loads [Lee et al. 2004; Ghosh et al. 2014; Salerno et al. 2004; Poncino and Macii 2006; Stanley-Marbell and Rinard 2016]. Thus, the total power consumption is approximately equal to the dynamic power:

$$P_{chan} = P_{dyn} = \alpha \cdot C_{tot} \cdot V_{DD} \cdot V_{swing} \cdot f \tag{1}$$

where C_{tot} is the total channel capacitance, accounting for driver, pin and wire contributions, V_{DD} is the driver supply voltage, and V_{swing} is the voltage swing between electrical levels. f is the clock frequency, and $\alpha \in [0, 1]$ is the *switching activity factor*, that accounts for the probability of occurrence of a value change in each clock cycle.

The vast majority of techniques for optimizing energy consumption in serial buses focuses on lowering α, reducing the number of intra-word and inter-word bit toggles by means of a proper *encoding* algorithm [Lee et al. 2004; Ghosh et al. 2014; Salerno et al. 2004; Ren et al. 2012; Zeng et al. 2014; Poncino and Macii 2006; Stanley-Marbell and Rinard 2016; Jahier Pagliari et al. 2016b]. We refer to the total sum of bit value changes during the operation of a bus as the *transition count* (TC).

Until recently, the literature on low-power bus encodings focused prevalently on parallel buses, due to the greater optimization opportunities offered by multiple wires. With the increase in use of serial interconnects, in the last few years, interest on low-power serial bus encodings has grown significantly [Lee et al. 2004; Ghosh et al. 2014; Salerno et al. 2004; Ren et al. 2012; Zeng et al. 2014; Poncino and Macii 2006; Stanley-Marbell and Rinard 2016; Jahier Pagliari et al. 2016b].

In general, encoding algorithms can be categorized into *lossless* or *accurate* encodings, for which in absence of channel errors the transmitted data can be decoded *exactly* at the receiver's end, and *lossy* or *approximate* encodings, which sacrifice total accuracy in the transmission, in order to reach larger TC reductions.

Lossless algorithms can be used in all types of I/O buses, i.e., data, addresses, and instructions. Most of them have been proposed in the context of Network on Chip (NoC) links [Lee et al. 2004; Ghosh et al. 2014; Salerno et al. 2004; Ren et al. 2012; Zeng et al. 2014], but they are also applicable to off-chip buses. A common characteristic of lossless approaches is to leverage *a priori assumptions* on the data to be encoded, in order to achieve TC reduction. Frequently, the data feature considered is *temporal correlation*, i.e., dependence among subsequent words. For example, the authors of [Lee et al. 2004] propose a simple scheme called SILENT based on *differential encoding* (DE), i.e., transmitting the bitwise difference between subsequent words; this solution allows to reduce TC when data are strongly correlated; however, when data are uncorrelated, pure DE can even lead to a power consumption *increase* [Lee et al. 2004].

Several other lossless encodings try to overcome this issue. The method proposed in [Ren et al. 2012], transmits either the bitwise difference between consecutive words or the one between every other word, depending on a prediction of data correlation. The authors of [Zeng et al. 2014], instead, propose to invert the bit values only in the even positions of a word, when the intra-word transition count is more than half the length of the word. In [Ghosh et al. 2014], TC reduction is achieved by swapping the

positions of some bits within words. All these three algorithms, however, suffer from the major drawback of requiring *redundant* bits, which are extremely costly in serial connections. In fact, these bits are transmitted either on a separate physical line, effectively doubling the cost of the connection, or on the same line as the data, reducing the available bus bandwidth. Thus, while for parallel buses redundancy is a powerful degree of freedom for data encoding, for serial buses it is basically unacceptable. Another lossless scheme [Salerno et al. 2004] is specifically targeted at serial display interfaces (e.g., DVI or TMDS), and exploits the locality of data in images; pixel *differences*, rather than pixel values, are sent on the bus, and a look-up-table is used to map the most common differences, i.e., the smallest ones, to low TC codewords.

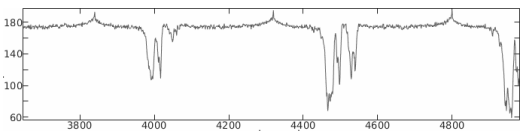
Lossy encodings, on the other hand, can be used only for *data* buses. The less general context of application is compensated by the larger achievable power savings at the expense of non-exact decoding. An early work in this field is found in [Poncino and Macii 2006], again in the context of display interfaces. The encoding relies on a smart rounding with bounded error of some LSBs of the transmitted pixels, which accounts also for inter-pixel transitions. More recently, the *Rake* algorithm [Stanley-Marbell and Rinard 2016] heuristically attempts to minimize the TC, under a maximum value deviation constraint. In practice, Rake approximates each word by inverting some of its bits to generate long sequences of ones and zeros. Interestingly, Rake does not exploit data correlation to reduce the TC. In [Jahier Pagliari et al. 2016b] the authors propose a lossy encoding named *Serial T0*, which targets a specific category of signals consisting of long phases of highly-correlated data, alternated with short phases of low correlation (*bursty* signals), typical of many realistic sensor inputs. Serial T0 approximates data (zeroing out the TC) during high correlation phases, based on the assumption that these do not convey much information.

3. MOTIVATION

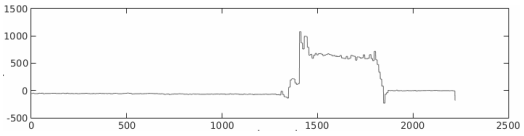
3.1. Characterization of Realistic Inputs

As pointed out in previous works [Jahier Pagliari et al. 2016b; Poncino and Macii 2006], data streams transmitted to/from devices such as sensors or peripherals are often characterized by a feature that we call *irregular temporal correlation*. By temporal correlation here we refer to the *similarity in magnitude* of consecutively transmitted data. The correlation is *irregular* because it is not always present; realistic streams typically consist of the alternation of relatively long regions with strong temporal correlation, and sporadic regions of faster and larger variations, in which correlation is very weak (called *bursty regions*). Some examples from real peripherals are shown in Figure 1. The graph in the upper left corner shows the serialized data from one of the three RGB color channels of a CCD camera. Correlated regions represent sections of the image in which the color is almost constant, whereas bursty regions (e.g., around the 4000-th word) represent lines or boundaries between two different colors. Similarly, the trace generated by each axis of an accelerometer, shown in the upper right graph, is strongly correlated when the device is idle or moving at a constant speed, while large variations occur correspondence of sudden movements (e.g., the interval between the 1300-th and 1800-th words). The lowermost graph shows a biomedical signal taken from an ECG, which exhibits a similar and well-known feature. Large variation areas correspond to heartbeats, and correlated areas to the interval between two pulses.

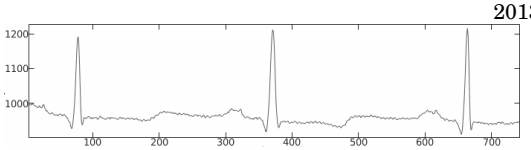
In the vast majority of the sensor and actuator signals available in public databases [Goldberger et al. 2000; Shoaib et al. 2013; Kodak ; OSR] correlated regions are much longer than bursty ones. Based on this observation, following the typical "common-case" paradigm for power optimization, an effective encoding for this type of data should reduce power especially in correlated regions. Incidentally, these regions



CCD-Camera/Display bus trace [Kodak].



Accelerometer (x-channel) bus trace [Shoaib et al. 2013].



ECG bus trace [Goldberger et al. 2000].

Fig. 1. Examples of correlated bus traces.

are also those in which error tolerance is higher; indeed, the core of the "information" is carried by bursty regions, and correlated regions can generally be approximated with less impact on the overall quality [Jahier Pagliari et al. 2016b].

3.2. Differential Encoding

When trying to reduce transition (and hence power) on serialized correlated data, Differential Encoding (DE) is a very effective solution in terms of efficiency/cost tradeoff [Lee et al. 2004]. DE consists of transmitting the *bitwise difference* between the current and previous words. In mathematical terms, codeword bits are generated as:

$$B_i[t] = b_i[t] \oplus b_i[t-1],\ \forall i \in [1, n] \tag{2}$$

where $b_i[t]$ is the i-th bit of the input word at time t, and $B_i[t]$ is the corresponding bit of the DE codeword. At the receiver, decoding is achieved by the following expression:

$$b_i[t] = B_i[t] \oplus b_i[t-1],\ \forall i \in [1, n] \tag{3}$$

DE is based on the observation that when data are represented in a positional notation (the most common choice for sensors and actuators), values that are numerically close have also a small *Hamming Distance* (HD) on average. This property is due to the fact that MSBs of numerically close values are in most cases equal. Exceptions occur when the two values are across a power of two (e.g., 00111111 and 0100000) or, for two's complement, when the two values are across zero, in which case the HD becomes large due to sign extension. However, both of these cases are statistically less likely.

Figure 2 shows the empirical Probability Mass Functions (PMF) of the HD for strongly correlated data. The graphs have been obtained generating 1 million 12-bit value pairs, of which the first one is selected randomly, and the second one is picked within a range of ± 25 from the first ($\pm 0.6\%$ of the full scale). The three PMFs have similar shape, showing that HD values ≤ 4 cover approximately 85% of the cases.

In practice, DE is effective because, when two words have small HD, the corresponding codeword B will contain a long string of 0s in the MSBs. Once serialized, this code will produce few level transition on the bus. Notice that DE is effective also for the cases in which the HD is *very large*, since in this case the codeword B consists of a long string of 1s in the MSBs, which still does not introduce transitions.

3.3. Limitations of DE

While DE is generally effective, it has an important limitation when used for typical data from/to sensors and actuators. Consider Figure 3, which shows the *qualitative* dependence between the average HD of the inputs and the TC of the corresponding DE codewords [Lee et al. 2004]. When the HD is about half of the word-length n, DE

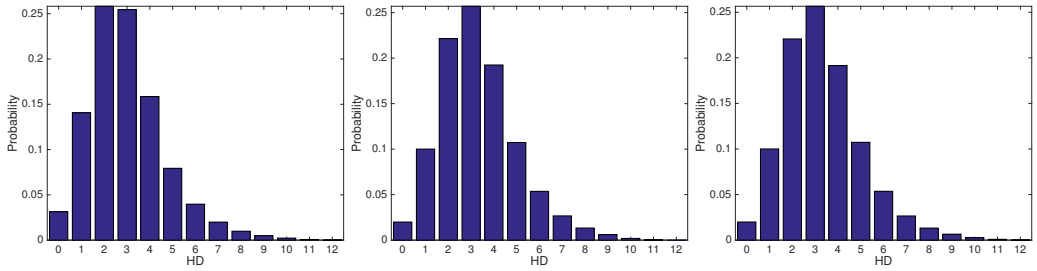


Fig. 2. HD PMF for 1 million pairs of random 12-bit words with absolute value difference $|w_1 - w_2| \leq 50$.

actually incurs in a power overhead, as the TC of codewords becomes larger than that of unencoded input data, which is approximately independent from the average HD.

Data transmitted from/to off-chip peripherals often have a relatively short bitwidth. This is true in particular for sensors, which normally sample data on 8-12 bits [Vahid and Givargis 2001]. Therefore, even when transmitting highly correlated data, such small bitwidths result in HD falling exactly in the "bad" range ($\frac{n}{3} < HD < \frac{2n}{3}$) for DE.

An example of this phenomenon is shown in Figure 4, which reports in blue the histogram of the HD obtained analyzing the serialized pixels of the RGB image *Lena* [Kodak]. Despite the good temporal correlation of the image pixels, 59% of samples have HD$\in [3:5]$. For these words, DE will not be able to reduce the TC on the bus.

A possible accurate solution to this issue has been devised in [Ren et al. 2012]. However, the proposed encoding requires redundant bits, which as explained in Section 2 add a large overhead to serial buses, either in terms of throughput or cost. A low-cost solution to the problem is Approximate DE, described in the next section.

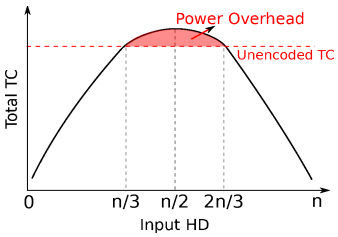


Fig. 3. Input Hamming Distance vs. Transition Count for DE.

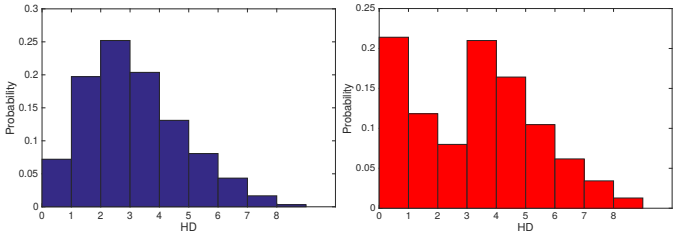


Fig. 4. PMF of the HD between consequent words in the bus trace produced by the *Lena* image, before and after LSB-saturation.

4. APPROXIMATE DIFFERENTIAL ENCODING

4.1. Approximate Differential Encoding

The limitations of DE described in the Section 3 can be overcome exploiting the error tolerance quality exhibited by many of the applications that involve serial links. We propose thus a *lossy* version of DE called Approximate DE (ADE). Its basic principle is to *reshape the HD distribution* of the input data to increase the probability of those HD values which are most favorable for DE. To do so, the basic form of ADE simply *saturates some LSBs of the input word to all 1s or all 0s*, before encoding the whole word with accurate DE. Thanks to saturation, the least significant part of the DE output will not introduce any transition, being composed of either all zeros or all ones.

This simple idea is very effective in reshaping the HD distribution. As an example, the red histogram of Figure 4 is obtained by saturating the 2 LSBs of each word in the trace generated by *Lena*. The distribution is visibly more skewed towards HD$= 0$, and 45% of word pairs have now $\frac{n}{3} < HD < \frac{2n}{3}$, a 14% less than in the original histogram.

The main parameter of ADE is the number l of saturated LSBs. Calling n the bitwidth of the input data, ADE operations are represented by the following equations:

$$\begin{aligned} B_i[t] &= b_{l+1}[t] \oplus b_{l+1}[t-1],\ \forall i \in [1,l] \\ B_i[t] &= b_i[t] \oplus b_i[t-1],\ \forall i \in [l+1,n] \end{aligned} \tag{4}$$

where $b_n[t]$ is the MSB and $b_1[t]$ is the LSB. The maximum error introduced by ADE for integer data representations can be computed as $E_{MAX} = 2^l - 1$. A pseudo-code implementation of ADE encoding is reported in Algorithm 1.

In the example of Figure 4, the saturation of 2 LSBs introduces a maximum error of: $2^2 - 1 = 3$, about 1.2% of the maximum value representable on 8-bit (255). As demonstrated visually in Section 7, this error produces a minimal reduction of quality in the decoded image. In spite of the limited error, ADE significantly reduces the number of transitions on the serial bus with respect to its accurate counterpart. For the *Lena* example, transitions are reduced of about 61%.

ALGORITHM 1: ADE encoding algorithm.

input : w: input word (of length n); l: number of saturated bits
output: B: codeword

```
/* Saturation Phase */
for i ∈ [1, l] do
|  t_i[t] = w_{l+1}[t]
end
for i ∈ [l + 1, n] do
|  t_i[t] = w_i[t]
end
/* Bitwise Difference Phase */
for i ∈ [1, n] do
|  B_i[t] = t_i[t] ⊕ t_i[t − 1]
end
```

It is worth emphasizing that, although it introduces the exact same amount of error, ADE is more flexible than a reduced-bitwidth serial transmission based on conventional DE, that simply avoids the transmission of the l LSBs. In fact, the amount of approximation introduced by ADE can be modified at runtime in an application-driven or system-driven way, without having to rely on a variable-width transmission. In other words, the number of saturated bits l can be set according to the application requirements or to the system status (e.g., state of charge of a battery in a mobile device), whereas the total bit-width n remains constant. This simplifies the transmitting and receiving circuits, and can be easily integrated with all standard serial protocols (e.g. *SPI*, *I2C*, etc.) without having to change the master/slave interfaces of existing peripherals. On the contrary, variable-width transmission is not only much more complex to implement from the point of view of transmitter and receiver complexity, but also impossible to integrate with legacy devices based on standard protocols. For example, SPI-based devices [Motorola 1989] often have *fixed-size* MOSI/MISO registers. Other protocols, such as I2C [NXP 2014], support variable-width transmission only in multiples of one byte, a too coarse granularity to support fine tuning of quality and energy. Finally notice also that the decoding part of ADE, which is identical to the one

of DE (Eq. 3) is independent from l, hence no synchronization between transmitter and receiver is needed when the accepted error changes.

4.2. Example of Operation

An example of operation of ADE is reported in Table I, for 8-bit unsigned binary data. The table shows the input data trace in the leftmost column pair, whereas ADE codewords are found on the rightmost set of columns. The two remaining column sets are included for sake of comparison, and report the outputs of conventional DE and LSB-saturation (LSBS) for the same inputs. In this example, both LSBS and ADE saturate 3 LSBs of each word, which corresponds to allowing a maximum relative error of 2.7% for 8-bit inputs. Columns labeled TC report, for each encoding, the transition count generated when the corresponding words are transmitted on a serial bus, assuming a LSB-first protocol and including also inter-word transitions. Column |E| reports the absolute error introduced by approximate encodings (LSBS and ADE) on each word.

This example shows how ADE performs better than both LSBS and ADE, by combining their features. The total transition count is reduced of $\frac{33-17}{33} \approx 48\%$, whereas the average absolute error per word is just $\frac{19}{10} = 1.9$, which corresponds to less than 0.75% of the full-scale value (255).

Table I. Example of operation of the ADE algorithm for a trace of 8-bit unsigned data.

Input		DE		LSBS			ADE		
Word	TC	Word	TC	Word	TC	|E|	Word	TC	|E|
00001011	4	00001011	4	00001000	2	3	00001000	2	3
00001111	2	00000100	2	00001111	2	0	00000111	2	0
00001101	4	00000010	2	00001111	2	2	00000000	0	2
00001101	4	00000000	0	00001111	2	2	00000000	0	2
00010111	4	00011010	4	00010111	4	0	00011000	2	0
00100011	4	00110100	4	00100000	2	3	00110111	4	3
00000100	2	00100111	4	00000111	2	3	00100111	4	3
00001101	4	00001001	4	00001111	2	2	00001000	2	2
00001110	2	00000011	2	00001111	2	1	00000000	0	1
00001011	3	00000101	3	00001000	2	3	00000111	1	3
Totals	33		29		22	19		17	19

4.3. Hardware Implementation

One of the advantages of ADE with respect to other approximate encodings is the simplicity of its hardware implementation. The encoding and decoding circuits for ADE can be realized as shown in Figure 5, in which both schematics refer to a 4-bit implementation for easier visualization.

The ADE encoder is composed of two main section, highlighted by dashed areas in Figure 5. The *saturation* section is simply an array of multiplexers, each of which determines if the corresponding bit in the input word (b_i) is forwarded to the next part, or if it is replaced with one of the bits of higher weight. The selection signals of the multiplexers are the bits of the ADE parameter l (see Eq. 4). For instance, if $l = 1$, i.e., we saturate one bit, $l_0 = 1, l_1 = 0$, and each MUX selects the second input form the top, resulting in $t_0 = b_1$, and $t_i = b_i, \forall i \geq 1$. For a generic n-bit word, the saturation netlist consists of n n-way multiplexers, each having $\log_2 n$ control inputs, in order to encode all possible values of $l = 0, \ldots, n-1$. This simple circuit implements LSB saturation, and is equivalent to lines 1-6 of Algorithm 1.

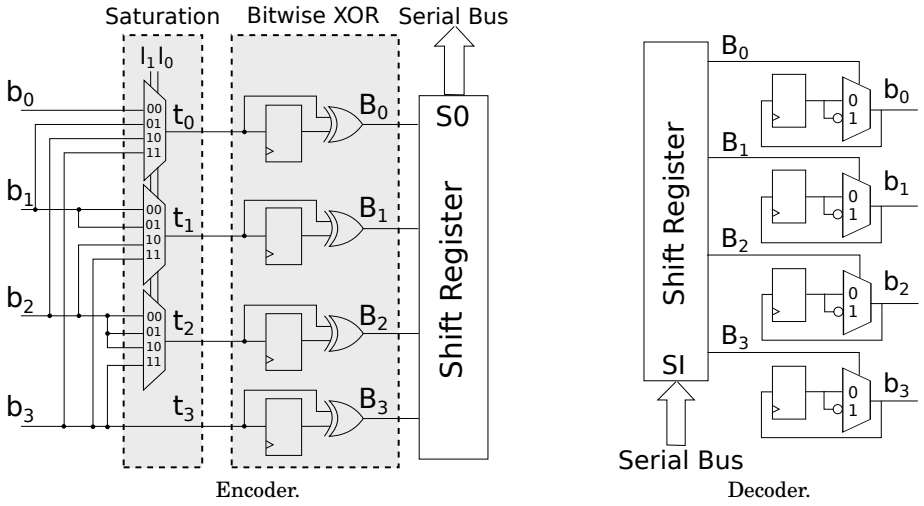


Fig. 5. Hardware implementation of 4-bit ADE.

The second section, labeled *bitwise XOR*, implements word-level Differential Encoding taking as input the intermediate saturated word. The previously transmitted datum is memorized in an array of flip-flops, and is used to perform the bitwise XOR between corresponding bits of the current and previous words. Finally, the result of the bitwise XOR phase is loaded into a shift register which acts as serializer.

The decoder is even simpler. At all times, the last decoded word is memorized in some flip-flops, initialized with zeros at the beginning of a transmission. When a new word is received and deserialized, a series of multiplexers selects between the memorized value and its negation, depending on the corresponding bit read from the parallel output of the shift register. As mentioned in Section 4.1, decoding is independent of l, which therefore does not need to be shared between transmitter and receiver.

Importantly, both encoder and decoder have a latency of a single clock cycle (as for accurate DE). Serialization and deserialization are not considered since they are present in any serial transmission. Moreover, both units have a throughput of one word per cycle, therefore they do not introduce any performance overhead in the communication. Their area and power consumption is also very limited, as they are composed of very simple logic and memory elements.

5. ADE VARIANTS AND IMPLEMENTATION DETAILS

5.1. Selective ADE for Small Variation Preservation

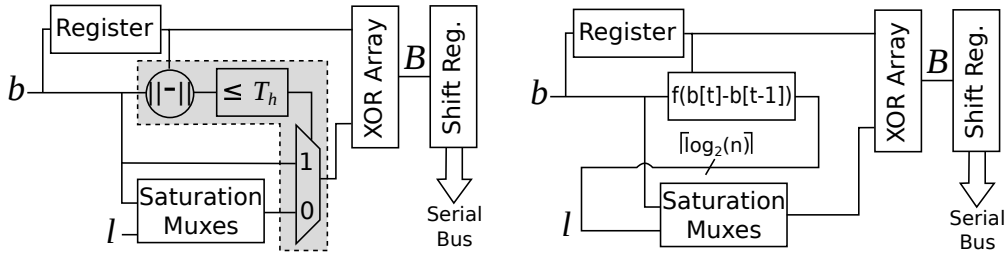
ADE approximates signals by eliminating small variations on the input data. However, there might be contexts in which small variations in a signal are important. This is particularly true when the amplitude of the signal is small. As an example, consider the reproduction of a digital audio recording of a speech. In presence of a large amplitude (high loudness), the noise resulting by LSB approximations can be tolerated because it does not impact the intelligibility of the spoken words. Conversely, when the loudness is low (e.g., the speaker moved away from the microphone), even a small noise might significantly undermine the quality of the reproduction.

Whenever small variations in the signal are important for a particular application, ADE can be slightly modified in order to preserve them more finely, at the price of reduced power saving, yet still higher than that of accurate DE. We call this variant *Selective* ADE (SADE). The basic idea behind this encoding is to only accept approxi-

mations (by means of LSB saturation) when the input signal varies significantly, and use accurate DE when the variation is small. In mathematical terms, SADE implements the following equation:

$$B[t] = \begin{cases} DE(b[t], b[t-1]) \text{ if } \|b[t] - b[t-1]\| \leq T_h \\ ADE(b[t], b[t-1], l) \text{ otherwise} \end{cases} \quad (5)$$

where $DE()$ and $ADE()$ are function-like representations of Eq. 2 and 4 respectively. A high-level block diagram of a SADE encoder hardware implementation is shown on the left in Figure 6, with the additional components with respect to basic ADE highlighted by the dashed area. The decoding circuit is exactly the same as DE and ADE.



Basic SADE. Generalized SADE or n-SADE.

Fig. 6. Hardware Implementations of Selective ADE encoders.

"Large" variations of the signals are detected by comparing two consecutive values against a threshold T_h. The latter must be set according to the application requirements, and is in principle independent from the number l of saturated bits of ADE. However, it is reasonable to set these two parameters jointly. In particular, setting:

$$T_h = K \cdot (2^l - 1) \quad (6)$$

corresponds to implicitly accept a maximum error equal to $K \cdot 100\%$ of the current variation of the signal. For example, if $K = 1$, the encoding saturates l bits (i.e., generates a maximum error of $2^l - 1$), only when the variation of the signal is larger than $2^l - 1$. In other words, the error is accepted when it alters the variation of the signal of less than 100%.

Within the class of possible SADE configurations for all values of T_h and l, ADE becomes a special case corresponding to $T_h = 0$. Thus, a software or hardware implementation of SADE can revert to ADE on-the-fly simply by setting the parameters appropriately. In Section 7 we will show that SADE is effective only when the input data trace has sufficiently long phases of low correlation. Conversely, when the input mostly consists of small variations, SADE never does approximations, and reduces to accurate DE.

The concept of SADE can be further extended, generalizing the relation between the amount of approximation and the local variation of the signal. A scheme of this generalized encoder is shown on the right in Figure 6. For example, instead of a "binary" decision between transmitting accurately or approximately, one could think of using *multiple thresholds*, each enabling a different degree of approximation (i.e., a different value of l). A specific variant of this scheme, which we call n-SADE, generalizes Equation (6) to n thresholds, where n is the bus data bitwidth. In practice, instead of setting a single value of l and a threshold T_h, as in basic SADE, quality is controlled by imposing a maximum ratio K between the variation of data and the allowed error.

For every new word to be encoded, the appropriate value of l is obtained from the local variation of the data, using K as proportionality coefficient:

$$l = \lfloor \log_2 \| E_{MAX} + 1 \| \rfloor = \lfloor \log_2 \| \frac{b[t] - b[t-1]}{K} + 1 \| \rfloor \tag{7}$$

It can be seen that Equation 7 can be implemented using n thresholds, corresponding to increasingly larger values of data variation, and consequently to an increasing number of saturated bits l.

Indeed, generalizations of SADE tend to increase the complexity in the encoding circuitry. For the particular case of n-SADE, the most general implementation requires n comparators, one for each threshold. Simpler implementations are possible, but only for specific values of the parameters (e.g., if K is a power of two). Despite this increase in complexity, the preliminary experiments that we performed on n-SADE did not show a consistent improvement in terms of power saving at iso-quality with respect to basic SADE. However, the number of possible relations between data variation and allowed approximation, and the corresponding implementations, are virtually infinite. A more in-depth analysis of SADE generalizations will be the subject of future work.

5.2. Dynamic Adaptation of Power/Quality Tradeoff

The main parameter of ADE encoding and its variant SADE is the number of saturated bits l, which influences the maximum error accepted by the encoding and should be set taking into account application-level constraints [Han and Orshansky 2013; Chippa et al. 2013]. The problem of finding the optimal l is essentially an error-constrained power minimization. The designer must set the *maximum* l that allows to meet application-level constraints on quality, resulting into the minimum power consumption on the bus. For many realistic applications (e.g., radio, multimedia, etc.), however, quality constraints are specified through statistical aggregate metrics, such as the Mean Absolute Error (MAE), or the Mean Square Error (MSE), rather than in terms of maximum error. ADE can be adapted to these metrics thanks to the fact that l can be easily changed at runtime. In particular, the parameter can be stored in a register, so that its value can be changed in a single clock cycle.

Since l only determines an *upper bound* on the error, the ideal way to meet a constraints in terms of statistical error metric (e.g. MAE) would consist in keeping track of the metric inside the encoder, computing the *actual* error introduced by saturation on each transmitted word. This value would then be compared with the application-level constraint, in order to calculate the new value of l for the next word. While this scheme would be feasible in software, it would introduce a significant overhead to a hardware implementation of the encoder.

A simpler and effective solution consists in periodically alternating between different values of l according to a fixed *duty cycle* (DC) computed offline. This solution works under the reasonable assumption that LSBs of input words are uniformly distributed [Landman and Rabaey 1995]. In this condition, it can be easily demonstrated that the saturation error is also a uniform random variable (e), taking values in the range between 0 and $E_{MAX}[l] = 2^l - 1$. Consequently the MAE and the MSE can be computed analytically as:

$$\begin{aligned} MAE[l] &= \mathbb{E}(e) = \frac{E_{MAX}[l]}{2} \\ MSE[l] &= \mathbb{E}(e^2) = \frac{E_{MAX}[l](2E_{MAX}[l]+1)}{6} \end{aligned} \tag{8}$$

Both equations only depend on E_{MAX}, which is in turn function of the encoder parameter l. Given a constraint on MAE or MSE, we simply need to invert the equations

to compute the *ideal* value of the maximum error $\hat{E}_{MAX}$ that would generate the desired value of the aggregate metric. In general, this value cannot be enforced by ADE with a single l, due to the fact that the parameter must be an integer, and therefore the achievable values for E_{MAX} are *quantized*. However, lower and upper bounds on l can be computed:

$$\begin{aligned} l_L &= \lfloor \log_2(\hat{E}_{MAX} + 1) \rfloor \\ l_H &= \lceil \log_2(\hat{E}_{MAX} + 1) \rceil \end{aligned} \tag{9}$$

Alternating between these two values with the proper duty cycle we can almost exactly achieve the imposed constraint. Supposing that the metric of interest is MAE, with target error $\hat{E}_{MAE}$, we must find a duty cycle $DC < 1$ so that:

$$DC \cdot MAE[l_H] + (1 - DC) \cdot MAE[l_L] = \hat{E}_{MAE} \tag{10}$$

Solving Equation 10 for DC yields:

$$DC = \frac{\hat{E}_{MAE} - MAE[l_L]}{MAE[l_H] - MAE[l_L]} = \frac{2\hat{E}_{MAE} - E_{MAX}[l_L]}{E_{MAX}[l_H] - E_{MAX}[l_L]} \tag{11}$$

In this equation, the denominator is always an integer, and represents the minimum period (in clock ticks) of the duty cycle. The numerator represents the number of clock cycles in each period for which the ADE encoder must use $l = l_H$, before switching to $l = l_L$ for the rest of the period. A similar relation can be computed for the MSE case.

The DC can in general only be approximated since (i) the numerator in Eq. 10 might be non-integer, and (ii) the trace might not contain an integer number of periods. Both conditions cause a slight bias in the error, which however can be forced to be on the conservative side (smaller MAE/MSE) by truncating the numerator of Eq. 10 (rather than round it) and by always starting the duty cycle with $l = l_L$.

Given the initial assumption of uniformity in the LSBs, the value of the DC can be computed offline. Then, the cost of this solution is simply that of a counter and a comparator, which increment and decrement l according to numerator and denominator of Eq. 11. Switching l has practically zero cost in the hardware implementation of ADE, as it simply consists in changing the value contained in a register.

5.3. Compatibility with Standard Serial Buses

The ADE family of encodings can be easily integrated with existing standard serial bus protocols. Most of those protocols, such as Serial Peripheral Interface (SPI) [Motorola 1989], Inter-Integrated Circuit (I2C) [NXP 2014], Controller Area Network (CAN) [ISO 2015] and Integrated Interchip Sound (I2S) [Philips 1996], specify an interface composed of three main types of wires: (1) one or more data lines, (2) a separate clock line, (3) one or more control lines (e.g., slave select in SPI). Obviously, ADE is applied only to data lines, whereas clock (which provides synchronization) and control lines (which have very low switching activities) are not encoded. None of the mentioned protocols specifies a particular semantic on data bits, which can therefore be freely encoded without violating the standard. Moreover, since ADE does not alter the bit-width of input words, our encoding can be fully integrated with preexisting interfaces that comply to one of these protocols.

The details on ADE usage depend on the protocol considered. In SPI, the entire control of the transmission takes place through the Slave Clock (SCLK) and Slave Select (SS) lines. Thus, the Master-Out-Slave-In (MOSI) and Master-In-Slave-Out (MISO) lines are *only* used for data bits, and can be fully encoded with ADE. The same applies also to the I2S protocol, in which the Serial Data (SD) line only transmit audio PCM

samples. In other cases, such as I2C, the data line is used, in different instants, to transmit both data and control information (e.g., slave addresses). ADE encoding and decoding must be applied selectively to data words, which are the only ones that can tolerate approximations, skipping control words. However, although with a reduced impact on the total savings, ADE is fully applicable also to these cases.

Other protocols, such as RS-232 [EIA 1969] ad Controller Area Network [ISO 2015] are asynchronous, i.e., timing information is conveyed through data lines and no separate clock lines are present. RS-232 transmits synchronization and control information through start/stop bits (and optional parity bits, etc) on the data line. CAN is a much more complex standard, which embeds up to 64 bytes of data in *frames* that contain synchronization, priority, acknowledge and CRC information. All these information must have deterministic values in order for the transmission to complete correctly. Nevertheless, ADE can still be used on data bits, whose semantic is not defined by the standards. The only case when ADE cannot be used straight-forwardly is when a standard specifies a precise data encoding. This is the case for some display protocols such as MIPI's DSI [MIPI 2004], which use Transition-Minimized Differential Signaling (TMDS). In this case, although ADE can still be applied in cascade, it is likely to have limited effectiveness.

Although most standard serial bus protocols (e.g. SPI, I2C, CAN, etc.) do not specify any constraint relative to DC balance [Webster and Eren 2014], our encoding can be easily modified to support this feature. By construction, ADE produces codewords that contain more 0s than 1s when input words are temporally correlated. However, it can be easily observed that the encoding would work exactly in the same way *if XOR gates were replaced by XNORs*. It is thus possible to devise a DC-balanced version of ADE called *ALternate* ADE (ALADE), in which words are encoded alternating XOR-ing and XNOR-ing words. In general, the balancing of 0s and 1s is achieved by slightly sacrificing power savings. The worst case is when the input trace is perfectly constant: with ADE, the encoded trace contains only 0s, and the TC reduces to 0. With ALADE, perfect DC balance is achieved by introducing 1 transition per word.

Hardware modifications to support ALADE are straight-forward, and omitted for brevity reasons.

6. ADC/ENCODER CO-OPTIMIZATION

When considering the bus that connects an off-chip sensor IC to a processing element, ADE can be co-optimized with the Analog-to-Digital Converter (ADC) contained in the sensor chip to achieve greater power savings. The family of converters for which this co-optimization is possible are Successive Approximation Register (SAR) ADCs [Manganaro 2011]. These relatively old converters have recently regained a lot of popularity for embedded applications, due to their low implementation cost, given by the fact that they imply a *single* chain of Sample and Hold (S/H) and threshold comparator [Manganaro 2011]. Nevertheless, their energy consumption can be comparable to that of a long off-chip bus; a modern commercial product can consume up to $17.3pJ/bit$ at maximum speed ($48MHz$ clock), for a total of $207.6pJ$ per 12-bit conversion [Maxim 2013].

A conceptual schematic of a SAR converter is shown in the dashed area of Figure 7. Internally, these ADCs employ a Digital-to-Analog Converter (DAC) to compare the input analog voltage sampled by the S/H ($V_{S/H}$) to increasingly accurate approximations (V_{APP}). Their functionality is based on dichotomic search. In the following we will briefly recall it in order to explain the interaction with ADE. Throughout the section, and without loss of generality, we assume that the format of the digital output is unsigned binary, and that the quantization of the analog voltages is uniform.

A new conversion is initiated, normally by a processor, by acting on the Start Of Conversion (SOC) control signal. Initially, the Successive Approximation Register is

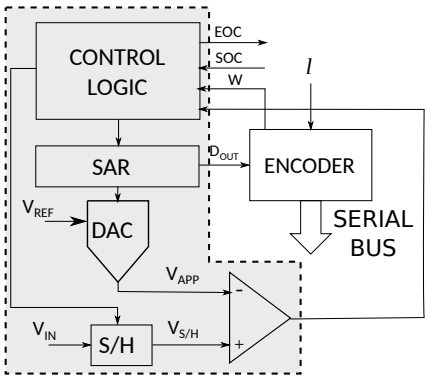


Fig. 7. Schematic of a SAR ADC and of its connection with an ADE encoder.

loaded with the value $10..0$, which after D/A conversion, corresponds to half of the full-scale voltage ($V_{APP} \approx \frac{V_{REF}}{2}$). This value is compared with $V_{S/H}$ by the threshold comparator. MSB of the SAR is kept at 1 and is reset to 0 otherwise. The rationale of this operation is that all and only the values greater than V_{APP} have a 1 at the MSB in their output binary representation.

In the successive conversion step, the second MSB of the SAR is set to 1, so that the register content is either $010..0$ or $110..0$ depending on the outcome of the previous step. These values correspond to either $\frac{V_{REF}}{4}$ or $\frac{3V_{REF}}{4}$ in the analog domain. Once again the approximation is compared with the sampled analog input, and the result is used to determine the final value of the second MSB of the SAR. This process is repeated a number of times equal to the desired output bit-width, since each step determines the value of one bit of the digital output, from MSB to LSB. Finally, the End-Of-Conversion (EOC) signal is activated to notify the rest of the system that the process has completed.

The latency of a SAR ADC is linear in the number of bits of the output; power consumption, instead, is approximately constant throughout the conversion, since the involved components are the same in each step. Consequently, the energy consumed by the converter is in first approximation linearly dependent on the output bit-width. This characteristic can be used in combination with the ADE family of encodings to reduce both conversion and transmission energy, with the scheme shown in Figure 7. Several variants are possible, depending on the integration between encoder and ADC.

Let us denote with n the maximum precision of the ADC. Supposing that transmitting at maximum precision is needed at some point during the system lifetime[1], n will also correspond the word-length on the serial bus.

A first way to combine encoder and ADC is to force a reduced bit-width A/D conversion using the saturation parameter l. It is an obvious waste of energy to perform A/D conversion on the entire bit-width n, since some LSBs will be ignored by the encoder because of saturation. We can therefore stop SAR A/D conversion earlier; as explained above, SAR converters process sample bits from MSB to LSB. Hence it is sufficient to halt the conversion after $n - l$ clock cycles, inducing an energy saving (and latency reduction) proportional to l. Most commercial SAR ADCs already support variable bitwidth operation (input W in Figure 7), so basic early stopping does not require internal modifications of the converter [Maxim 2013]. The l parameter can be simply subtracted from the maximum bitwidth and forwarded to the ADC as $W = n - l$.

[1]Otherwise the choice of the ADC was overestimated.

A further reduction of power can be achieved if a true co-design of ADC and of the encoder is possible. In fact, the SAR can be easily modified to internally perform LSB-saturation; in this way, the ENCODER block in Figure 7 can become a simple DE encoder (array of XOR gates), reducing complexity and avoiding useless redundancy.

Finally, an even tighter integration of SAR ADCs is possible when using Selective ADE. SADE performs saturation only when the difference between two consecutive words is larger than a threshold T_h; the comparison between these two words requires that SAR A/D conversion *always* starts using the maximum bit-width n. However, since SAR ADCs compute MSBs first, it is possible to determine whether the difference is greater than T_h *during* the conversion. In this case, the conversion can stopped after $l - n$ bits, otherwise it must be carried out with maximum accuracy.

In particular, let $b_{1:n-l}[t]$ be the current sample being digitized by a SAR after $n - l$ clock cycles (in which the $n - l$ MSBs are correct, and the remaining l LSBs are 0s). Let $\hat{b}_{1:n-l}[t]$ be the word with the same MSBs as $b_{1:n-l}[t]$ but with l 1s at the LSBs. The final SAR output at the n-th clock cycle is guaranteed to be in the range between these two words. Let $b[t-1]$ be the last fully converted word. A conversion can be stopped early if either one of these two conditions verifies:

$$\begin{aligned} &b[t-1] \geq \hat{b}_{1:l-n}[t] + T_h \text{ or} \\ &b[t-1] \leq b_{1:l-n}[t] - T_h \end{aligned} \tag{12}$$

The implementation of these comparisons obviously requires additional hardware with respect to the SADE encoder shown in Figure 6. However, this scheme allows to tune the precision (and energy consumption) of the SAR ADC on a sample-by-sample basis.

7. EXPERIMENTAL RESULTS

7.1. Experimental Setup

For the assessment of the effectiveness of ADE, we selected a set of realistic inputs traces from error resilient applications involving serial buses [Han and Orshansky 2013]. A first group of test data contains the traces generated by three common sensors found in mobile devices, namely, accelerometer, magnetometer, and gyroscope, obtained from the Pervasive Systems Research Data Sets [Shoaib et al. 2013]. For all sensors we considered two different use cases, namely *walking* and *biking*. The decimal data have been converted to binary format, using the datasheets of commercial products to match the specific parameters (bit-width, full scale ranges, etc.) [NXP 2016; Freescale 2013; STM 2012]. All three datasheets refer to devices equipped with serial interfaces, using either I2C or SPI.

A second group of data contains electrocardiogram (ECG) samples taken from the Physionet online database [Goldberger et al. 2000], as representatives of biomedical applications. To simulate serial transmission, ECG samples have been converted to unsigned binary on 11-bit, according to the specifications of Physionet.

Multimedia peripherals are also typically interfaced with computing elements via serial protocols (e.g. SPI, MIPI, etc.) To mimic these data we considered serial traces produced by voice recordings and images. Audio data have been obtained from the Open Speech Repository [OSR], and consist of 16-bit two's complement Pulse Code Modulation (PCM) samples. Images have been taken from the Kodak Database [Kodak] and are represented as 24-bit RGB, i.e., 8-bit unsigned per each color channel. Table II summarizes all the realistic input sets that we used and the corresponding data representations.

We compare ADE against three state-of-the-art approximate encodings, namely LSBS [Poncino and Macii 2006], Rake [Stanley-Marbell and Rinard 2016] and Serial-T0 [Jahier Pagliari et al. 2016b], as well as against accurate DE [Lee et al. 2004].

Table II. Realistic input datasets used for experiments.

Input Set	Format	Bit-Width	Ref.
Accelerometer	Two's complement	12-bit	[NXP 2016]
Magnetometer	Two's complement	16-bit	[Freescale 2013]
Gyroscope	Two's complement	16-bit	[STM 2012]
ECG	Unsigned	11-bit	[Goldberger et al. 2000]
Audio	Two's complement	16-bit	[OSR]
Images	RGB (Unsigned)	3·8-bit	[Kodak]

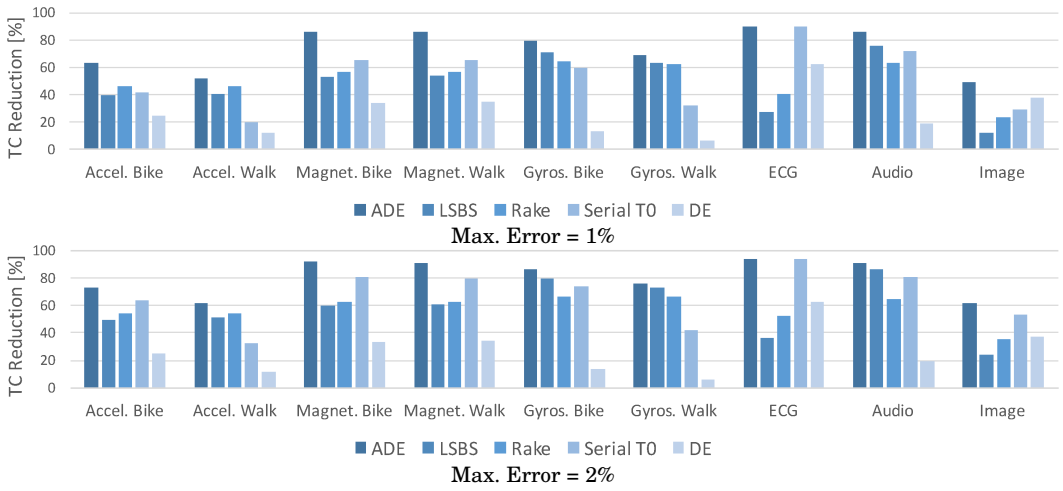


Fig. 8. TC reduction of ADE and state-of-the-art encodings for all considered input data sets, and for two different maximum relative error constraints.

For a fair comparison, we consider only irredundant competitors. In fact, redundant encodings require either a higher cost (due to additional lines) or a higher signaling frequency (due to header bits), in order to transmit data with the same throughput. In addition to that, because of the small bit-width of the considered traces, additional transitions introduced by redundant bits often nullify power gains on data bits. Indeed, for several inputs, redundant approaches [Ren et al. 2012; Ghosh et al. 2014] obtain worse results compared to simple DE.

As done by the authors of competitor encodings, we assume without loss of generality that the underlying bus protocol does not transmit control bits on the data line. Protocols such as SPI or I2S belong to this category.

7.2. Comparison of Transition Count

The first set of experiments evaluates ADE versus the other encodings in terms of transition count (TC). A more accurate comparison accounting for hardware overheads will be presented in Section 7.4.

7.2.1. TC reduction versus maximum error. Figure 8 reports the percentage TC reduction achieved by the various encodings for the chosen input data sets. The two charts refer to two constraints in terms of maximum allowed relative error (E_R), with respect to each input set's Full Scale range (*FS*), specifically, $E_R = 1\%$ and $E_R = 2\%$.

The parameters of the different encodings are set so as not to violate the error constraints. For instance, the l parameter in ADE is computed as $l = \lfloor \log_2(\frac{E_R}{100} \cdot FS + 1) \rfloor$. Parameters of other encodings are set in a similar way.

Figure 8 shows that ADE obtains better TC reduction with respect to all other encodings for 8 out of 9 of the considered input sets, regardless of the maximum allowed error. ADE is only slightly outperformed by Serial T0 [Jahier Pagliari et al. 2016b] for ECG data, because of the strong *bursty* nature of these traces, that perfectly matches the working principle of Serial T0. However, the latter becomes less effective when burstiness is less evident such as in traces generated by gyroscope and accelerometer.

In some specific conditions, e.g., accelerometer and magnetometer inputs generated by the biking activity with $E_R = 1\%$, the TC reduction of ADE is about 20% larger than that of every other encoding. Moreover, by construction, ADE always obtains better TC reductions compared to accurate DE. This is an important property, that justifies the presence of approximations in ADE, and that is not always guaranteed by other encodings. For example, DE outperforms one or more approximate solutions for the image and ECG inputs.

Input image ADE output

Fig. 9. ADE image transmission example ($E_R = 2\%$, $l = 2$).

In order to get the feeling of the impact of approximations in ADE, Figure 9 shows an example on the 24-bit RGB image *Lena*. The original image is shown on the left, whereas the one on the right is the result of ADE encoding and decoding, with E_R set to 2% (corresponding to $l = 2$). Such configuration yields a reduction of 61% of TC, yet without differences noticeable by the human eye. This example shows that $E_R = 2\%$ is an acceptable constraint for realistic error resilient applications, and therefore gives value to the previous results. The visual similarity is also confirmed when considering numerical metrics; in fact, the Mean Structural SIMilarity (MSSIM) index [Wang et al. 2004] between the two images is 99.85%.

7.2.2. TC Reduction versus Actual Mean Error. Another way to compare different approximate encodings is to consider the tradeoff between TC reduction and actual mean error on decoded data. As a matter of fact, the parameters of all considered encodings affect the *maximum* error; however, the actual error introduced for each data word is in general smaller than this bound, and, by aggregation, also the final *mean* error is in general smaller than the imposed constraint.

Figure 10 reports the tradeoff between mean output error and TC reduction for all encodings and for the same traces of Figure 8. The points on the Pareto-curves relative to ADE have been obtained varying the l parameter from $l = 1$ to $l = 5$. For a fair comparison, the parameters of other approximate encodings have been set to allow the exact same maximum error. In particular, LSBS uses the same parameter l as ADE,

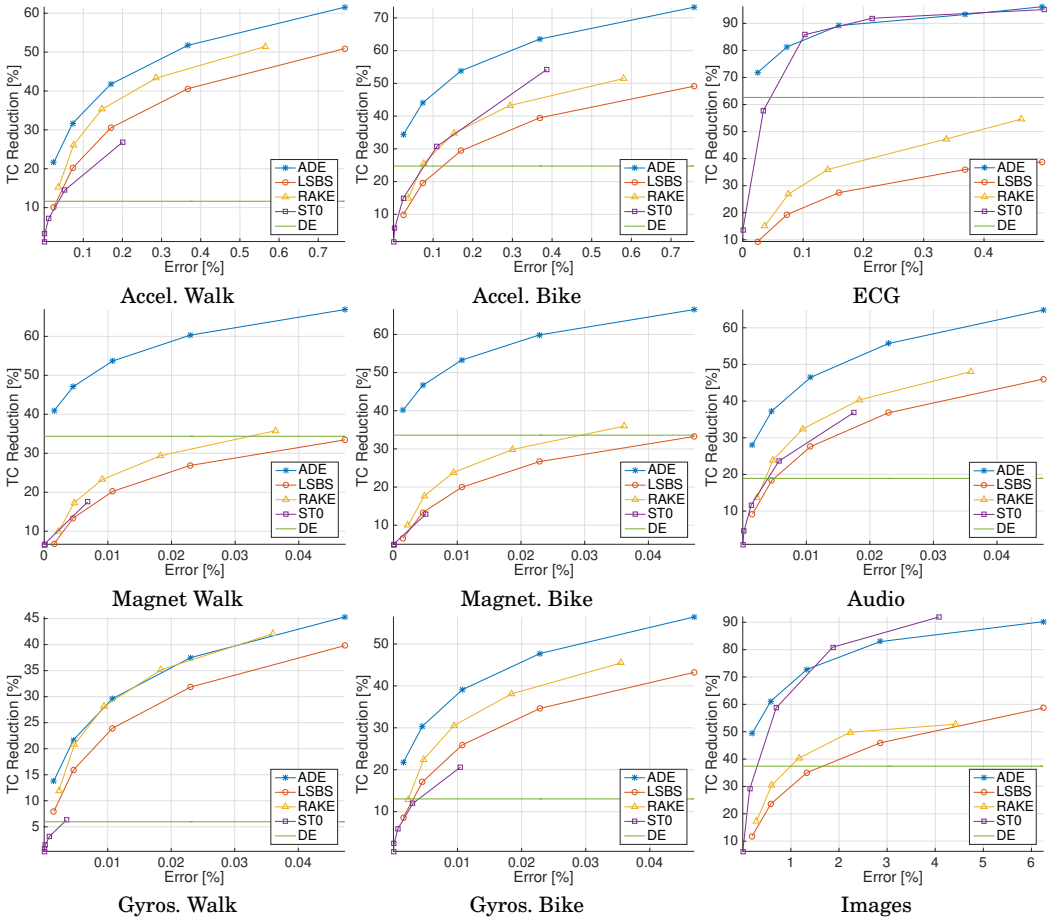


Fig. 10. Comparison between ADE and state-of-the-art encodings in terms of Transition Count reduction versus Mean Error tradeoff, for all considered input sets.

while Rake and Serial-T0 directly use a maximum error threshold, which has been set to $2^l - 1$ for each value of l. The horizontal line represents the TC reduction of DE, which since this encoding is accurate, is constant with respect to the error axis.

The curves of Figure 10 show that, even when considering the actual mean error, ADE outperforms all other encodings for the majority of the inputs. The top-left corner of the graphs represents optimality, i.e., an ideal encoding that reduces transitions to zero while not introducing errors. In most cases ADE *dominates* other encodings in the Pareto sense, by reducing the transitions more for a given error.

It is worth observing that the relations between the TC reductions of different encodings in these graphs are not the same as in Figure 8. On one hand, this is because the considered quality metric is different. On the other hand, differences are also due to the error ranges considered. In fact, by varying l (and the other parameters) in a fixed interval $[1:5]$, we are considering different ranges of average relative error, depending on the bit-width of the data (see the x-axes of the graphs).

7.2.3. Effectiveness for Random Data. ADE and the other encodings are built specifically to deal with error resilient traces. Nonetheless, it is interesting to assess their effectiveness for random data. Figure 11 shows the TC reduction obtained by these en-

codings when applied to 12-bit unsigned inputs generated randomly according to a Gaussian distribution. Each point in the graph refers to a different set of Gaussian inputs. All sets have the same mean μ, equal to half of the full scale range (2^{11}), whereas the standard deviation σ is varied in increasing powers of 2, from 2^0 to 2^{12}. Each input set consists of 10,000 words. The maximum allowed error for lossy algorithms has been set to 1% of the full scale range.

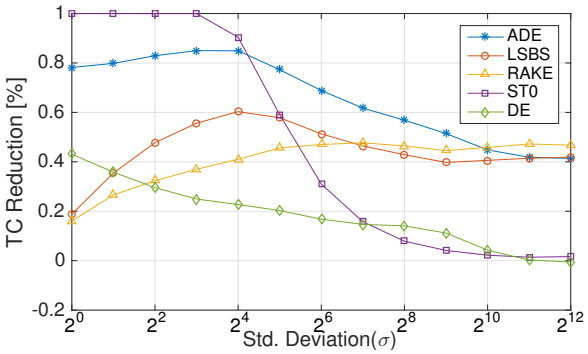


Fig. 11. TC reduction of ADE and state-of-the-art encodings for random Gaussian data traces with different standard deviations (σ) and same mean (μ), with maximum error $E_R = 1\%$.

Figure 11 shows that ADE is the best encoding in terms of TC reduction for quite a large range of standard deviations (between 2^5 and 2^9). Expectedly, when σ is close to 0, Serial T0 dominates the other encodings; a small σ corresponds in fact to *strongly correlated* data, which represents the ideal case for Serial T0. However, the effectiveness of this encoding decreases dramatically for larger σ. ADE, conversely, which is not based on thresholds, has a more gradual decrease in TC reduction for larger σ's.

At the other extreme of the graph, i.e., for highly-uncorrelated data traces, ADE yields approximately the same TC reduction as LSBS. This happens because DE has no effect for this kind of data, and actually incurs a slight *increase* in TC. Such behavior is expected, since a Gaussian distribution with large standard deviation tends to approach a uniform distribution, which is the worst-case condition for DE. In fact, it can be proven that, for uniform data, the HD between consecutive words has a *binomial* distribution centered in $n/2$, with n being the input bit-width [Stan and Burleson 1995]. However, thanks to saturation, ADE is still able to reduce the TC of about 40%.

In general, ADE guarantees good reductions for a wide range of σ, and dominates for intermediate ones, which are the most common in real applications. Notice that ADE is the *only* lossy encoding whose approximations are justified in all conditions, since by construction it is never dominated by an irredundant accurate encoding such as DE.

7.3. Transition Count Reduction for ADE Variants

In this section we focus on the two main variants of our encoding: Selective ADE (SADE) and ALternate ADE (ALADE).

Figure 12 reports the TC reduction obtained for ADE and its variants, as well as for accurate DE, for a subset of the inputs described in Section 7.1. In this experiment, the l parameter of ADE and its variants has been set to $l = 4$, which corresponds to a maximum allowed error of 15. The bar chart reports three variants of SADE, characterized by different values of the threshold: $T_h = 16$ (SADE-16), $T_h = 32$ (SADE-32) and $T_h = 64$ (SADE-64). As explained in Section 5.1, these settings correspond to a maximum allowed error circa 100%, 50% and 25% of the signal variation.

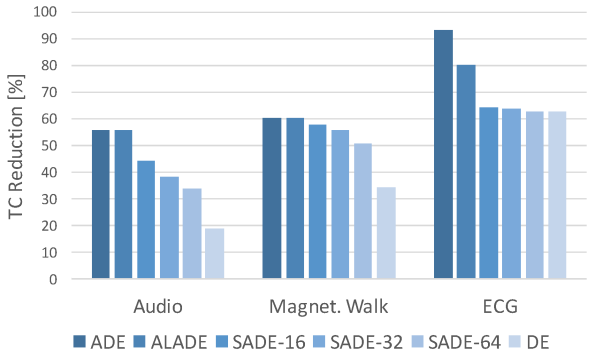


Fig. 12. TC reduction of DE, ADE and its variants (with $l = 4$) for a selection of input sets.

The first observation is that ALADE, achieves for audio and magnetometer inputs, practically the same TC reduction as ADE (55.7% versus 55.6% and 60.3% versus 60.0% respectively). This indicates a small impact of inter-word transitions, which are the only source of difference in TC between ADE and ALADE. For ECG data, the difference in TC is more marked. In fact, these inputs have long almost constant phases in the intervals between two heart beats, which tend to fall into the worst-case condition for ALADE discussed in Section 5.3.

As expected all variants of SADE reduce the TC less with respect to basic ADE, since some approximations are *avoided* in order to preserve small variations in the signal. In general, a larger T_h generates smaller toggle reductions, yet always larger than those of accurate DE. The audio and magnetometer traces are composed by the alternation of small variation and large variation phases, while in ECG data, small variations are predominant. Thus, as for ALADE, SADE variants obtain better results for the first two data sets.

It is important to underline that both ALADE and SADE do not aim at improving the total TC reduction obtained by basic ADE. On the contrary, these variants allow to apply the principles of ADE also to situations in which the original encoding would not be usable (e.g., when DC balancing is required), or would not produce good quality results (e.g., when small variations are important). Therefore, the results of Figure 12, in which ADE dominates its variants in terms of *pure* TC reduction, are to be expected.

A better way to evaluate the effectiveness of SADE is to use an error metric that takes signal variation into account. To this end, we consider a metric that normalizes the error on each word with respect to the local variation of the data. In principle, this could be achieved simply dividing the error by the absolute value difference between two consecutive words. However, when two consecutive words are equal, such metric would produce a division by zero. Therefore, we use a *windowed error* metric E_w, in which the error on each word is normalized to the maximum variation of the signal in a sliding window of size $w \geq 1$, where w is computed as the minimum that avoids divisions by zero. Normalized errors on different words are then averaged to obtain the final aggregate metric. Mathematically, E_w is computed as:

$$E_w = \frac{1}{N} \sum_{t=1}^{N-w} \frac{\|b_d[t] - b[t]\|}{\max\limits_{t \leq s \leq t+w}(b[s]) - \min\limits_{t \leq s \leq t+w}(b[s])} \tag{13}$$

where $b[t]$ and $b_d[t]$ are the input and decoded words at time t respectively, and N is the length of the input trace. As desired, this metric gives more "weight" to errors happening in correspondence of small signal variations.

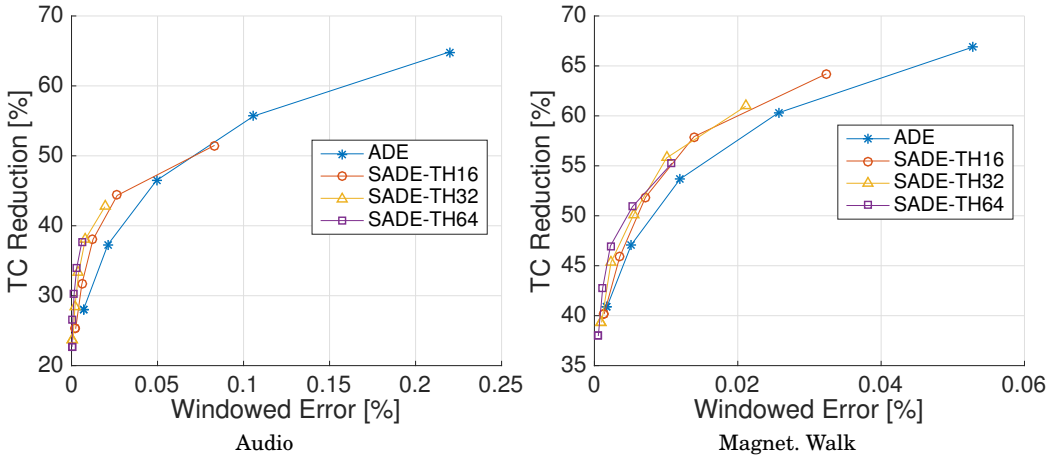


Fig. 13. E_w versus TC reduction tradeoff for ADE and SADE.

Figure 13 shows the error versus TC reduction tradeoff for ADE and SADE considering E_w, for audio and magnetometer inputs. Different points in the graph have been obtained varying the general parameter l of all encodings from $l = 1$ to $l = 5$. The window length for error computation is $w = 4$ for audio data, and $w = 5$ for magnetometer. When using E_w SADE outperforms ADE for both inputs. Notice that SADE variants with a larger T_h always produce very small windowed errors, regardless of the value of l, because they have less opportunities for approximations. However, when very high quality is required, SADE-64 configures as the best solution for both input sets. When slightly lower quality is acceptable, an intermediate solution such as SADE-16 might be preferable. In general, the choice of the appropriate T_h must be evaluated depending on the specific inputs considered.

Finally, we present a motivating example of the use of *duty cycling* in ADE to fine-tune the error, using a realistic application. In multimedia, image quality is often measured by the Peak Signal-to-Noise Ratio (PSNR), defined as the ratio between the maximum possible value of a signal and the square root of the MSE introduced by a lossy algorithm. Acceptable PSNR values, e.g., in the context of lossy compression algorithms, are in the range $20 - 40\ dB$ [Salomon 2006]. Consider an application that uses ADE to transmit image pixels, e.g., digitized by a camera. Assume that the PSNR of decoded data with respect to the input is required to be $\geq 38\ dB$, in order, for instance, to render the effect of approximate encoding negligible with respect to a following compression phase. Considering the 8-bit per color representation of image pixels, this requirement can be converted to $MSE \leq 10.306$.

Table III. Example of l-duty-cycling for image data.

l	**MSE**	**PSNR [dB]**	**TC Reduction [%]**
2	3.48	42.71	61
3	17.01	35.82	73
Duty Cycling	9.929	38.162	66

Table III shows the result of a simulated ADE transmission with $l = 2$ and $l = 3$ of the *Lena* image [Kodak]. With $l = 2$, the MSE is significantly smaller than the requirement, meaning that ADE is not fully exploiting the available error tolerance, and is loosing opportunities to reduce the TC. On the contrary, with $l = 3$, the requirement

Table IV. Synthesis results for 8/16-bit ADE and Serial T0 encoders and decoders on 45nm CMOS.

Circuit	8-bit version			16-bit version		
	Area [μm^2]	**Power @ 200MHz** [W]	**Power vs Freq.** [W]	**Area** [μm^2]	**Power @ 200MHz** [W]	**Power vs Freq.** [W]
ADE Encoder	85.02	$2.33 \cdot 10^{-5}$	$1.14 \cdot 10^{-13} f + 5.94 \cdot 10^{-7}$	173.57	$4.83 \cdot 10^{-5}$	$2.35 \cdot 10^{-13} f + 1.21 \cdot 10^{-6}$
ADE Decoder	68.80	$3.32 \cdot 10^{-5}$	$1.64 \cdot 10^{-13} f + 5.24 \cdot 10^{-7}$	136.53	$6.70 \cdot 10^{-5}$	$3.30 \cdot 10^{-13} f + 1.05 \cdot 10^{-6}$
Serial T0 Encoder	304.82	$4.59 \cdot 10^{-5}$	$2.20 \cdot 10^{-13} f + 1.95 \cdot 10^{-6}$	720.06	$1.11 \cdot 10^{-4}$	$5.22 \cdot 10^{-13} f + 6.29 \cdot 10^{-6}$
Serial T0 Decoder	73.38	$1.66 \cdot 10^{-5}$	$8.08 \cdot 10^{-14} f + 4.84 \cdot 10^{-7}$	146.41	$3.28 \cdot 10^{-5}$	$1.59 \cdot 10^{-13} f + 9.01 \cdot 10^{-7}$

in terms of MSE/PSNR is violated. Since l must be integer, no intermediate values are possible. Consequently, in this example, *no single value* of the parameter would allow to obtain an output PSNR that is superior to the imposed constraint, yet very close to it, in order to maximize power savings. To overcome this limitation, we can resort to duty cycling. Using equations similar to Eq. 10-11, but for the MSE metric, we obtain a period of 84 clock cycles, in which for the first 44, data is encoded with $l = 2$, and for the remaining 40 with $l = 3$. The result of encoding *Lena* with l-duty-cycling is reported in the last line of Table III. As shown, this solution is able to match the desired quality constraint almost perfectly. By doing so, it allows to improve the TC reduction of an extra 5% with respect to the fixed solution with $l = 2$. Notice that the MSE of the duty cycling solution is slightly *smaller* than the requirement, due to the conservative management of errors described in Section 5.2.

7.4. Hardware Overheads and Break-even Length Analysis

Sections 7.2 and 7.3 compared different encodings in terms of pure TC reduction. However, to fully evaluate the effectiveness of an energy-efficient bus encoding algorithm, a more detailed analysis must take into account the actual energy consumption on the bus, including overheads due to encoding and decoding circuitry.

To this end, we first synthesized ADE encoder and decoder (with the architectures of Figure 5) to assess their cost in terms of silicon area and power. The two architectures were specified at RTL using VHDL, and synthesized using Synopsys Design Compiler K-2015.06 on a 45nm standard cell library from ST Microelectronics. We verified the correctness of the post synthesis netlists by means of a timing simulation in Mentor Questa Simulator v10.4, with random input stimuli. During these simulations we also annotated the internal switching activity of the encoder and decoder netlists, which we then loaded in Synopsys PrimeTime J-2014.12 to accurately estimate power.

Table IV shows the results of synthesis for 8-bit and 16-bit versions of the codecs, using a $200MHz$ frequency target for both cases. The table also provides a simplified power versus frequency model, where leakage power is constant, and dynamic power linearly depends on frequency.

For sake of comparison, we also synthesized under identical conditions the encoding and decoding hardware of Serial T0, described in [Jahier Pagliari et al. 2016b]. Although both algorithms result in extremely low hardware overheads, the total power consumption of ADE is approximately 80% of that of Serial T0, for the 16-bit implementation at 200MHz, and 90% for the 8-bit version, denoting also a better scalability. A comparison with Rake [Stanley-Marbell and Rinard 2016] was not possible due to

Table V. Analysis of energy consumption and break-even line length for ADE and Serial T0 with realistic parameters.

Data Set	N	f [MHz]	V_{BB} [V]	Encoding	α	$E_{HW,W}$ [J]	$E_{LINE,W,LEN}$ [J/m]	l_{BE} [m]
Audio	16	4.0	1.8	**Unencoded**	0.339	-	$7.44 \cdot 10^{-10}$	-
				ADE	0.182	$1.13 \cdot 10^{-12}$	$3.98 \cdot 10^{-10}$	$3.27 \cdot 10^{-3}$
				Serial T0	0.299	$2.48 \cdot 10^{-12}$	$6.57 \cdot 10^{-10}$	$2.86 \cdot 10^{-2}$
Images	8	184.3	1.6	**Unencoded**	0.524	-	$4.54 \cdot 10^{-10}$	-
				ADE	0.142	$2.83 \cdot 10^{-13}$	$1.23 \cdot 10^{-10}$	$8.57 \cdot 10^{-4}$
				Serial T0	0.216	$3.14 \cdot 10^{-13}$	$1.87 \cdot 10^{-10}$	$1.18 \cdot 10^{-3}$

the fact that the paper did not provide hardware architectures for the codecs. However, in the case of Rake, it is evident from the relative complexity of the two algorithms, that the resulting circuitry would be significantly more complex than ADE.

Having the codec overhead available, we can finally evaluate the total energy of a serial transmission using ADE. To do so, we model the bus data line as a microstrip on a FR-4 PCB [Montrose 1998]. We assume a line with $25\mu m$ of width and $10\mu m$ of thickness, over a substrate of $200\mu m$ height with permittivity $\varepsilon_r = 4.5$. These parameters yield a capacitance per unit length equal to $C_{LEN} = 42.3\, pF/m$ [Asuni]. This value can be used with the model of Eq. 1 to evaluate the power (or the energy) per unit length consumed by the line. The other parameters of Eq. 1, depend on the application considered and are summarized in Table V. We assumed $V_{DD} = V_{Swing}$, and that encoder, decoder and serial line all use the same clock signal. The switching activity α and the bitwidth of input words N considered in this experiment refer to the transmission of the *Audio* and *Images* inputs of Section 7.1.

For both inputs, we compare the energy cost of transmitting unencoded data, with that of transmitting ADE codewords encoded with $l = 2$, and Serial T0 codewords obtained setting the error threshold to 7 [Jahier Pagliari et al. 2016b].

To do so, we compute the energy $E_{HW,W}$ necessary for ADE and Serial T0 codecs to process a single data word, using the power consumption data from Table IV. Moreover, we also compute the average energy per unit-length consumed by the microstrip to transmit one word ($E_{LINE,W,LEN}$). The latter depends on the encoding used, as both ADE and Serial T0 reduce the number of transitions (and thus α) on the line. These energy values take into account that both hardware implementations perform encoding and decoding in a single clock cycle, whereas the transmission of a word on the microstrip takes N clock cycles.

The last column of Table V reports the *break-even length* l_{BE} of the bus required to amortize the cost of the codec. An encoding saves energy only if the bus exceeds l_{BE}, because the benefits of reducing the TC become larger than the overheads due to the codec. For both inputs, the break-even length of ADE is in the order of one millimeter (0.85mm for *Images* and 3.27mm for *Audio*), while Serial T0 has larger l_{BE} (1.18 and 28mm, respectively). The large difference for the audio data is due both to ADE's more efficient codec and larger TC reduction. Considering that normal PCB lines are in the order of some centimeters long, this is a very good result, which proves the effectiveness of ADE in reducing the total energy consumption of a serial communication, and its applicability even to very short traces.

8. CONCLUSIONS

The energy consumption per transmitted word in long off-chip serial connections can be comparable to the execution of an instruction in an ultra low-power processor.

Therefore, energy optimization of serial interconnections is a very important design goal for modern embedded systems.

In this paper we have shown that Approximate Differential Encoding (ADE) and its variants (SADE, ALADE, etc.) are effective techniques to achieve this goal. Thanks to their simplicity, ADE and its variants can be implemented both in software and in hardware with extremely low overheads. Nevertheless, these solutions are very effective in leveraging the correlation properties of error resilient data in order to reduce the number of transitions on a serial bus, achieving results that are superior to approximate state-of-the-art encodings.

The combination of the effectiveness and simplicity of ADE result in significantly better total power savings with respect to its competitors, when all overheads are considered.

REFERENCES

Peter Alfke. 2008. *Xilinx Design Hints: Printed Circuit Board Design Considerations*.

Nicola Asuni. *PCB Impedence and Capacitance Calculator*. https://technick.net/public/code/cp_dpage.php?aiocp_dp=util_pcb_imp_microstrip

Wouter Bracke, Robert Puers, and Chris Van Hoof. 2007. *Ultra Low Power Capacitive Sensor Interfaces* (1st ed.). Springer Netherlands.

V.K. Chippa, S.T. Chakradhar, K. Roy, and A. Raghunathan. 2013. Analysis and characterization of inherent application resilience for approximate computing. In *Proceedings of the 50th ACM / EDAC / IEEE Design Automation Conference (DAC)*. 1–9.

EIA. 1969. *Interface between data terminal equipment and data communication equipment employing serial binary data interchange.* Electronic Industries Association: Engineering Department.

Freescale. 2013. *Xtrinsic MAG3110 Three-Axis, Digital Magnetometer*. Freescale Semiconductor.

S. Ghosh, P. Ghosal, N. Das, S.P. Mohanty, and O. Okobiah. 2014. Data Correlation Aware Serial Encoding for Low Switching Power On-Chip Communication. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. 124–129. DOI:http://dx.doi.org/10.1109/ISVLSI.2014.48

A.L. Goldberger, L.A.N. Amaral, L. Glass, J.M. Hausdorff, P.Ch Ivanov, R.G. Mark, J.E. Mietus, G.B. Moody, C.K. Peng, and H.E Stanley. 2000. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation* 101(23) (2000), e215–e220.

Jie Han and M. Orshansky. 2013. Approximate computing: An emerging paradigm for energy-efficient design. In *Proceedings of the 18th IEEE European Test Symposium (ETS)*. 1–6. DOI:http://dx.doi.org/10.1109/ETS.2013.6569370

N. Ickes, D. Finchelstein, and A.P. Chandrakasan. 2008. A 10-pJ/instruction, 4-MIPS micropower DSP for sensor applications. In *Proceedings of the IEEE Asian Solid-State Circuits Conference, 2008. A-SSCC '08*. 289–292. DOI:http://dx.doi.org/10.1109/ASSCC.2008.4708784

ISO. 2015. *ISO 11898-1:2015, Road vehicles – Controller area network (CAN)*. International Organization for Standardization.

D. Jahier Pagliari, E. Macii, and M. Poncino. 2016a. Approximate Differential Encoding for Energy-Efficient Serial Communication. In *Proceedings of the 26th Edition ACM Great Lakes Symposium on VLSI (GLSVLSI '16)*. ACM, 421–426. DOI:http://dx.doi.org/10.1145/2902961.2902974

D. Jahier Pagliari, E. Macii, and M. Poncino. 2016b. Serial T0: Approximate Bus Encoding for Energy-efficient Transmission of Sensor Signals. In *Proceedings of the 53rd Annual ACM / EDAC / IEEE Design Automation Conference (DAC '16)*. Article 14, 6 pages. DOI:http://dx.doi.org/10.1145/2897937.2898089

Kodak. *Image Database*. http://r0k.us/graphics/kodak/

P.E. Landman and J.M. Rabaey. 1995. Architectural power analysis: The dual bit type method. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 3, 2 (June 1995), 173–187. DOI:http://dx.doi.org/10.1109/92.386219

Kangmin Lee, Se-Joong Lee, and Hoi-Jun Yoo. 2004. SILENT: serialized low energy transmission coding for on-chip interconnection networks. In *Proceedings of the IEEE/ACM International Conference on Computer Aided Design ICCAD-2004.* 448–451. DOI:http://dx.doi.org/10.1109/ICCAD.2004.1382618

G. Manganaro. 2011. *Advanced Data Converters*. Cambridge University Press.

Maxim. 2013. *MAX11102/03/05/06/10/11/15/16/17 2Msps/3Msps, Low-Power, Serial 12-/10-/8-Bit ADCs. Datasheet*. Maxim Integrated.

MIPI. 2004. *Display Interface Specifications*. MIPI Alliance. http://mipi.org/specifications/display-interface

Mark I. Montrose. 1998. *EMC and the Printed Circuit Board: Design, Theory, and Layout Made Simple*. Wiley-IEEE Press.

Motorola. 1989. *MC68HC11 Reference Manual*.

NXP. 2014. *I2C-bus specification and user manual*. NXP Semiconductors.

NXP. 2016. *MMA8452Q, 3-axis, 12-bit/8-bit digital accelerometer*. NXP Semiconductors.

OSR. *Open Speech Repository*. http://www.voiptroubleshooter.com/open_speech/index.html

Philips. 1996. *I2S bus specification*. Philips Semiconductors.

M. Poncino and E. Macii. 2006. Low-energy RGB Color Approximation for Digital LCD Interfaces. *IEEE Transactions on Consumer Electronics* 52, 3 (Aug. 2006), 1004–1012. DOI:http://dx.doi.org/10.1109/TCE.2006.1706500

Xianglong Ren, Deyuan Gao, Xiaoya Fan, and Jianfeng An. 2012. Adaptive Low-Power Transmission Coding for Serial Links in Network-on-Chip. *Procedia Engineering* 29 (2012), 1618 – 1624. DOI:http://dx.doi.org/10.1016/j.proeng.2012.01.183 2012 International Workshop on Information and Electronics Engineering.

S. Salerno, A. Bocca, E. Macii, and M. Poncino. 2004. Limited intra-word transition codes: an energy-efficient bus encoding for LCD display interfaces. In *Proceedings of the 2004 International Symposium on Low Power Electronics and Design. ISLPED '04.* 206–211. DOI:http://dx.doi.org/10.1109/LPE.2004.1349337

David Salomon. 2006. *Data Compression: The Complete Reference*. Springer.

M. Shoaib, H. Scholten, and P. J. M. Havinga. 2013. Towards Physical Activity Recognition Using Smartphone Sensors. In *Proceedings of the 10th IEEE International Conference on Ubiquitous Intelligence and Computing and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*. 80–87. DOI:http://dx.doi.org/10.1109/UIC-ATC.2013.43

M. R. Stan and W. P. Burleson. 1995. Bus-invert coding for low-power I/O. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 3, 1 (March 1995), 49–58. DOI:http://dx.doi.org/10.1109/92.365453

Phillip Stanley-Marbell and Martin Rinard. 2016. Reducing Serial I/O Power in Error-tolerant Applications by Efficient Lossy Encoding. In *Proceedings of the 53rd Annual ACM / EDAC / IEEE Design Automation Conference (DAC '16)*. ACM, Article 62, 6 pages. DOI:http://dx.doi.org/10.1145/2897937.2898079

STM. 2012. *A3G4250D MEMS motion sensor: 3-axis digital output gyroscope*. ST Microelectronics.

Frank Vahid and Tony Givargis. 2001. *Embedded System Design: A Unified Hardware/Software Introduction* (1st ed.). John Wiley & Sons, Inc., New York, NY, USA.

Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (April 2004), 600–612.

John G. Webster and Halit Eren. 2014. *Measurement, Instrumentation, and Sensors Handbook* (2nd ed.). CRC Press.

Jian Zeng, Jian-Yang Zhou, and Rung-Bin Lin. 2014. Transition inversion coding with parity check for off-chip serial transmission. In *Proceedings of the 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. 634–637. DOI:http://dx.doi.org/10.1109/ICECS.2014.7050065